éducAlgo



Manuel d'utilisation

26 juin 2011

Table des matières

1 Tâche à effectuer : écrire un algorithme

Le logiciel *éduc*`Algo` propose une liste d'exercices dans un menu en haut de la page. L'utilisateur doit commencer par en choisir un (par défaut, aucun exercice n'est sélectionné).

Chaque exercice demande d'écrire un algorithme (c'est-à-dire un programme) qui, à partir de certaines données d'entrée, permet d'obtenir des résultats de sortie. L'écriture doit s'effectuer uniquement à partir des outils proposés à l'écran, en respectant une certaine syntaxe.

Exemple

Par exemple, choisissons l'exercice dont le titre est :
Somme des N premiers entiers.

Ce choix s'effectue dans le menu des exercices. L'énoncé apparaît :
Soit un nombre entier $N \geqslant 1$
Calculer et afficher S, la somme des nombres entiers de 1 à N :
$S = 1 + 2 + ... + N$

La première difficulté est de bien comprendre cet énoncé. Pour cela, il est recommandé d'imaginer d'abord des exemples simples. Lorsque N vaut 3, par exemple, les N premiers nombres entiers non nuls sont $1, 2$ et 3, et leur somme S vaut 6, car c'est le résultat de l'opération $1 + 2 + 3$.

La formule générale de S est en fait $S = 1 + 2 + \cdots + N$. Le plus pratique serait bien sûr de pouvoir écrire cette formule directement, et ce serait terminé. Mais les outils proposés ici ne le permettent pas, parce que, avec les contraintes de l'informatique, il est difficile d'obtenir la généralité, la souplesse et l'abstraction du langage mathématique. D'autre part il s'agit d'un environnement de découverte et d'initiation, qui se concentre sur les notions de base de l'algorithmique.

Dans ce cas, des connaissances mathématiques pourraient être utiles.

On peut démontrer en effet que $S = \dfrac{N \times (N + 1)}{2}$. Mais les expressions offertes par le logiciel pour cet algorithme ne permettent pas d'écrire cela.

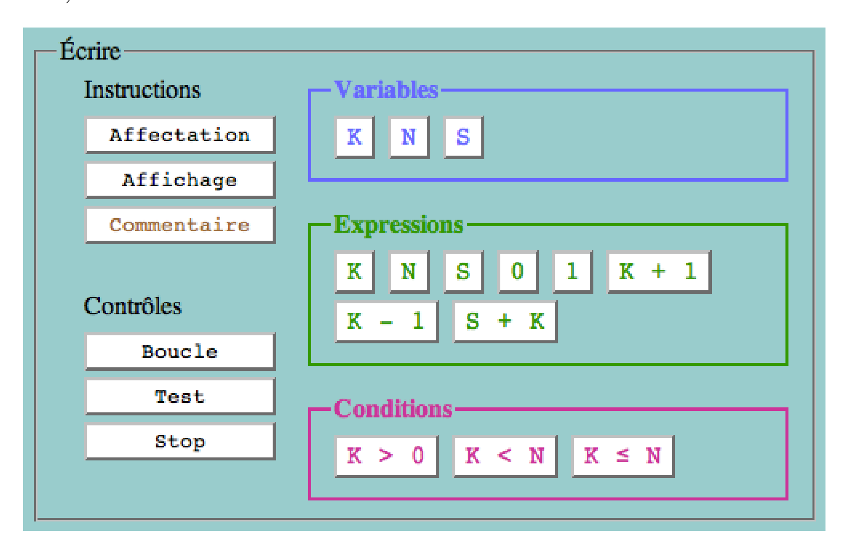
Nous allons voir dans la suite quels sont les outils disponibles ici et ce qu'ils permettent.

Remarque : La version la plus récente de *éduc*`Algo` propose dans le panneau *Écrire* des expressions supplémentaires qui permettent de résoudre le problème par la définition d'algorithmes différents. On ne décrit dans ce document qu'une seule solution et on laisse l'utilisateur découvrir les autres.

2 Comment écrire un algorithme ?

2.1 Avec quoi écrit-on ? Avec les boutons d'écriture

Le logiciel fonctionne un peu comme un traitement de textes, mais il n'utilise pas le clavier de l'ordinateur. Pour écrire quelque chose, on doit cliquer sur un des boutons d'écriture affichés à l'écran dans le cadre libellé *Écrire*, qui sont classés par catégories : *Instructions, Contrôles, Variables, Expressions, Conditions.*



Ces catégories correspondent à des concepts d'algorithmique :

– les *Variables* contiennent des valeurs, et leur contenu peut être changé au cours du temps. Ce sont des mémoires de travail.
– les *Instructions* disent à l'ordinateur quelles actions il doit effectuer. Par exemple, une *Affectation* consiste à changer la valeur d'une variable. La formulation en français serait « affecter une valeur à une variable », ou « mettre une valeur dans une variable ».
  Les formulations usuelles dans les langages informatiques ressemblent à $V = X$, où V est le nom de la variable et X une expression qui désigne une valeur. La variable est écrite à gauche et la valeur à droite. Le symbole d'égalité est utilisé, mais ce n'est pas l'égalité au sens mathématique. On peut trouver cela regrettable, mais c'est l'usage dans de très nombreux langages depuis des dizaines d'années et il vaut mieux s'habituer à cette notation.
  Dans le logiciel, nous proposons la formulation « `Poser V = X` ».
– les *Contrôles* indiquent à l'ordinateur dans quel ordre et à quelles conditions il doit effectuer les instructions (on dit aussi souvent *instructions de contrôle* ou *structures de contrôle*)

Par exemple, pour calculer V, la valeur absolue de N :

```
Si N > 0 faire :
    Poser V = N
  sinon faire :
    Poser V = - N
```

Le logiciel écrit les instructions avec des retours à la ligne et des décalages (on dit des *indentations*) pour mettre en évidence la structure. Le mot « faire » rappelle qu'il s'agit bien d'actions qui modifient l'état physique d'une machine (le contenu de la mémoire V change).

- les *Expressions* permettent de calculer et de désigner des valeurs.
  Par exemple $2N + 1$
- les *Conditions* permettent d'exprimer des propriétés concernant les valeurs et de savoir si ces propriétés sont vraies ou fausses à l'instant où on les évalue.
  Par exemple $N > 0$ est une condition *vraie* si N est strictement positif et *fausse* sinon.

Contrairement à un langage de programmation général, les variables, les expressions et les conditions n'ont pas à être créées de toutes pièces à partir de concepts généraux plus élémentaires. Ici, pour chaque exercice, le logiciel propose seulement un petit nombre de variables, expressions et conditions prédéfinies permettant de construire un algorithme solution (et éventuellement plusieurs). Cela allège les problèmes de syntaxe et fournit des indications sur des solutions possibles. Cela évite aussi de parcourir les 1000 pages d'un manuel d'un langage pour se demander quelle vont être les quelques 5 ou 6 instructions qu'on va utiliser parmi les centaines qui existent.

Cependant, les structures de contrôle de l'algorithme sont à mettre en place sans indications.

2.2 Où écrit-on ? A la position du curseur

Le fait de cliquer sur un bouton d'écriture provoque l'écriture d'un texte dans la zone de texte. D'une manière un peu analogue à un traitement de texte, cette écriture se produit à l'endroit où est placé le *curseur*, matérialisé par une marque. Dans un traitement de texte, cette marque est souvent un trait vertical clignotant, du type « | ». Ce n'est pas le cas ici, parce que l'écriture est soumise à certaines règles de syntaxe que le logiciel connaît et qui sont indiquées dans le curseur, pour aider l'utilisateur.

Le curseur est matérialisé par un rectangle jaune contenant une indication sur la nature de ce qui est attendu à cet endroit.

Lorsque le curseur est « | », on attend une instruction ou un contrôle ou rien. C'est un endroit où on peut insérer une instruction ou un contrôle, mais

on peut aussi ne rien écrire et passer à la ligne suivante en cliquant sur le curseur qui prend la forme d'une flèche « ↓ » lorsque la sourit le survole.

Le curseur peut aussi se transformer en sélection et contenir les indications suivantes : « *Variable?* », « *Expression?* », « *Condition?* ». Dans ce cas, on attend un élément de la nature indiquée, cet élément est obligatoire et doit être fourni à l'aide des boutons de droite correspondants.

Une fois qu'on a écrit quelque chose, le curseur passe à la prochaine position d'écriture, en indiquant ce qu'on attend à cet endroit.

Par exemple, si on veut écrire `Poser V = X`, il faut cliquer d'abord sur le bouton « Affectation », puis sur la variable « V », puis sur l'expression « X ».

Attention à une erreur fréquente : il se peut qu'une même lettre comme X apparaisse à la fois comme variable et comme expression. Quand on écrit `Poser X = 1`, il s'agit de la variable X, alors que dans `Poser N = X`, il s'agit de l'expression X.

C'est la même lettre, mais elle n'est pas interprétée de la même façon. Dans le premier cas (variable X), elle désigne un *contenant* (la case mémoire qui s'appelle X) alors que dans le second (expression X) elle désigne un *contenu* (la valeur qui est contenue dans la case mémoire de nom X).

Contrairement au langage mathématique, le symbole = employé ici n'est pas symétrique. C'est regrettable, mais c'est un usage tellement répandu en informatique qu'il vaut mieux s'y habituer. Il s'agit d'un autre langage, avec ses conventions propres.

2.3 Affichage des résultats

Le bouton « Affichage » permet d'insérer dans l'algorithme une instruction qui affiche la valeur de l'expression qui suit.

Pour permettre une présentation plus lisible des résultats, la valeur de l'expression peut être précédée d'un texte facultatif.

```
Afficher : S
```

Par défaut, il n'y a pas de texte mais simplement un deux-points.

Un clic sur le « : » ouvre un dialogue qui permet éventuellement de modifier le texte qui précède l'expression. Par exemple :

```
Afficher Somme = S
```

En cas d'erreur, on peut à nouveau cliquer sur le texte pour le modifier.

2.4 Commentaire

Le bouton « Commentaire » permet d'insérer dans l'algorithme un commentaire, c'est à dire une instruction qui ne fait rien.

On clic sur le bouton « Commentaire » ouvre un dialogue de saisie de texte qui contient par défaut le commentaire suivant : (Rien)

On peut à ensuite modifier le texte du commentaire avec un clic sur le texte affiché pour ouvrir à nouveau le dialogue qui permet de modifier le texte du commentaire. Par exemple : (Bla Bla)

3 Comment corriger ?

Il est quasiment inévitable d'avoir à modifier ce qu'on a écrit pour corriger ou compléter l'algorithme.

3.1 Comment déplacer le curseur ? Avec les touches « ↓ » et « ↑ » du clavier

La touche « ↓ » déplace le curseur vers le bas , à la ligne suivante. Quand le curseur atteint la fin du texte de l'algorithme, il continue en revenant au début de l'algorithme.

La touche « ↑ » déplace le curseur vers le haut, à la ligne précédente. Quand le curseur atteint le début du texte de l'algorithme, il continue en revenant à la fin de l'algorithme.

On peut aussi faire descendre le curseur en cliquant dessus.

Remarque : Parfois, sur certains navigateurs les touches flèches du clavier ne fonctionnent pas, ou sur certains claviers d'ordinateurs elles ne sont pas disponibles. Une alternative consiste à utiliser la touche « A » pour monter et la touche « Z » pour descendre.

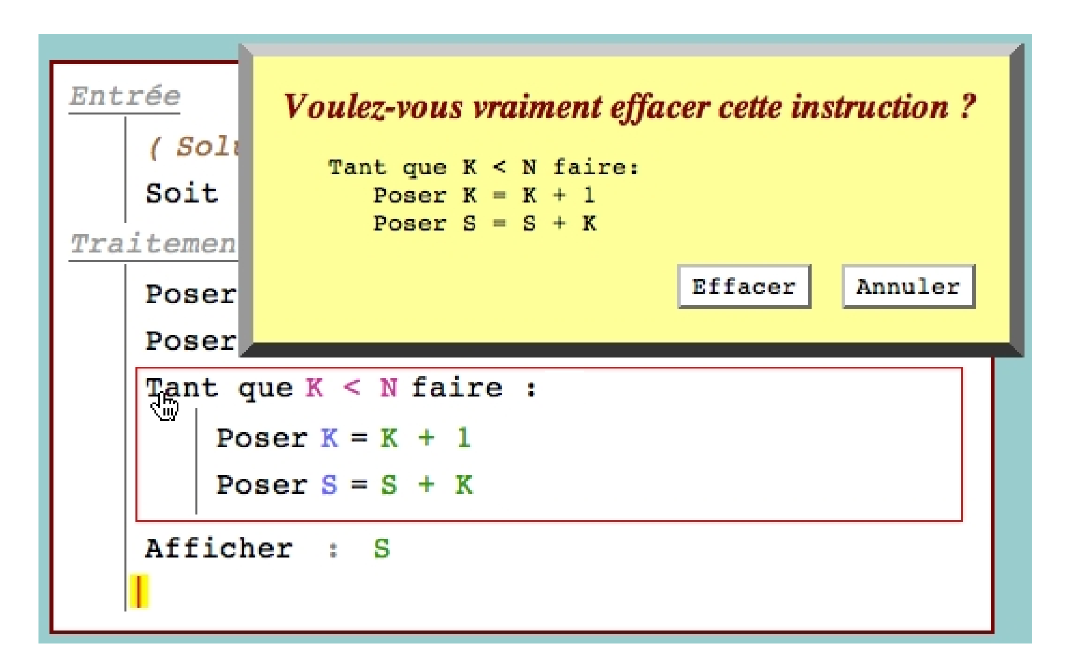
3.2 Comment modifier une variable, une expression ou une condition ?

Il suffit de la sélectionner dans le texte de l'algorithme, puis de cliquer sur une autre variable, expression ou condition parmi les boutons d'écriture.

3.3 Comment effacer une instruction ou un contrôle ? Clic dans une zone encadrée en rouge

Lorsque la souris survole les mots-clés d'une instruction, celle-ci est encadrée de rouge. On peut ainsi visualiser facilement les instructions simples, mais aussi l'imbrications éventuelles des listes d'instructions dans les contrôles.

Lorsque l'utilisateur clique une zone encadrée en rouge, un dialogue est ouvert qui demande à l'utilisateur de confirmer s'il a vraiment l'intention de supprimer l'instruction affichée, comme dans l'exemple ci-dessous :



Si la suppression est confirmée, l'instruction est effacée et le curseur se place à l'endroit où était l'instruction.

3.4 Résumé des actions disponibles sur l'interface

Les boutons d'écriture insèrent les éléments correspondant à la position du curseur selon les règles d'écriture.

De nombreuses autres actions de la souris sont possibles dans la zone d'édition de l'algorithme.

Déplacements du curseur

On peut déplacer le curseur avec les touches du clavier : « ↑ » ou « A » pour faire monter le curseur et « ↓ » ou « Z » pour faire descendre le curseur.

On peut cliquer sur une fin de ligne pour provoquer le déplacement du curseur à la ligne suivante :

Poser K = K + 1 ↵ Poser S = S + K ↵	Poser K = K + 1 ↵ Poser S = S + K ↵	Poser K = K + 1 ↵

On peut cliquer sur le curseur pour le faire descendre à la ligne suivante :

Poser K = K + 1 ↵ Poser S = S + K ↵	Poser K = K + 1 ↵ Poser S = S + K ↵	Poser K = K + 1 ↵ Poser S = S + K ↵

Sélections Effacement

On peut sélectionner *variables*, *expressions* ou *conditions* pour effectuer éventuellement ensuite une modification avec les boutons d'écriture :

```
Poser S = S + K ↵        Poser S = S + K ↵
```

On peut cliquer sur un *commentaire* une *instruction* ou un *contrôle* encadré en rouge, pour éventuellement l'effacer après confirmation :

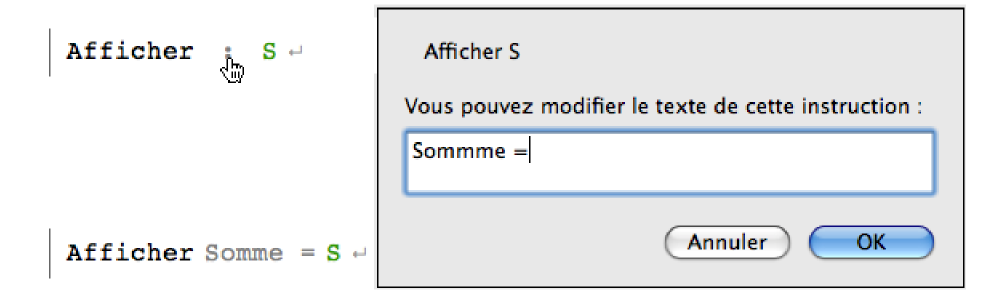
```
Tant que K < N faire : ↵          Tant que K < N faire : ↵
    Poser K = K + 1 ↵                 Poser K = K + 1 ↵
    Poser S = S + K ↵                 Poser S = S + K ↵
```

On peut aussi cliquer sur le symbole ↻ situé à la suite du menu de sélection d'exercice pour effacer toute la partie *Traitement* de l'algorithme.

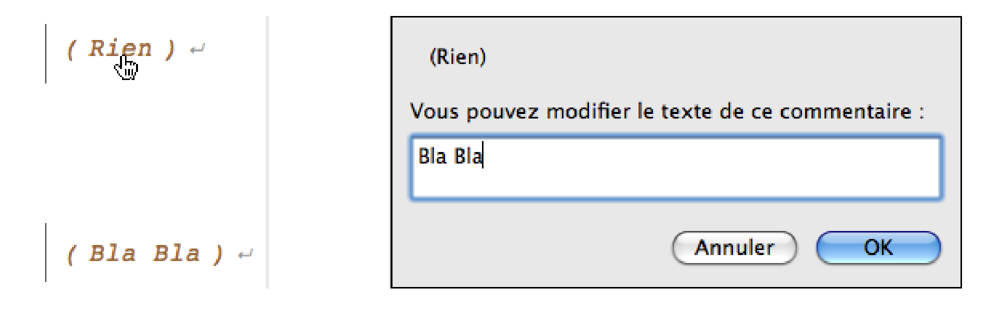
La touche d'effacement du clavier peut aussi être utilisée.

Modifications des textes

On peut modifier le message de l'étiquette d'une instruction d'affichage avec un clic sur le « : » ou sur l'étiquette :



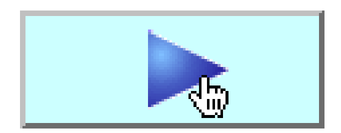
On peut modifier le texte d'un commentaire avec un clic sur le texte :



4 Exécution et Trace

4.1 Exécution de l'algorithme

Une fois l'algorithme rédigé, on peut cliquer sur le bouton *Exécuter*.



Pour chaque instruction de lecture exécutée, un dialogue s'ouvre pour demander à l'utilisateur de taper une valeur. Cette valeur doit être cohérente avec ce qui est attendu pour la variable correspondante, et qui est spécifié dans l'énoncé et dans l'instruction de lecture elle même.

S'il y a plusieurs variables à lire dans la partie *Entrée*, le dialogue de lecture affiche un rappel des valeurs des variables lues précédemment.

Lorsque les calculs sont terminés, une zone de résultats s'ouvre pour montrer ce qui a été lu et ce qui a été affiché par l'algorithme de l'utilisateur. Puis la solution correcte attendue pour les données qui ont été fournies est ensuite affichée

4.2 Trace de l'exécution

Si le résultat obtenu par l'utilisateur est différent de celui attendu, il est alors possible, pour obtenir le détail des calculs effectués à chaque étape par l'algorithme, de cliquer sur l'un des deux boutons de trace :

Montrer le déroulement de l'exécution *Montrer les valeurs des variables*

Le premier bouton affiche une description du déroulement de l'algorithme sous forme d'un tableau dont les colonnes décrivent pour chaque instruction :
- le *Calcul* à effectuer
- le *Résultat* de ce calcul
- l'*Action* effectuée pour modifier l'état des variables de l'algorithme

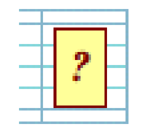
Le second bouton affiche un tableau qui montre les valeurs des variables pendant le déroulement de l'algorithme ;
- sur chaque ligne figure l'instruction exécutée
- suivie des valeurs de chaque variable, juste après son exécution.

Ces tableaux permettent d'analyser le fonctionnement de l'algorithme et de vérifier s'il correspond bien à ce qui est attendu.

5 Aide

Quand l'utilisateur survole avec la souris des différents éléments de l'interface, des messages d'aide apparaissent en haut à droite de l'écran. Il est utile de les lire avec attention lors de la prise en main de l'application.

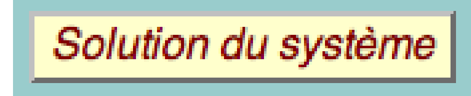
De plus, un clic dans cette zone d'aide quand elle contient un point d'interrogation « ? » affiche un mode d'emploi.





Si l'écriture de l'algorithme vous pose des problèmes,
éduc`Algo` propose selon les exercices :

– soit un bouton qui fournit une solution



– soit un menu qui propose des conseils pour créer l'algorithme



Tests

Pour certains algorithmes, il peut arriver que des jeux de valeurs particulières, pour les données en entrées, provoquent des situations critiques.

Pour aider à la mise au point de ces algorithmes, on trouve parfois dans le menu d'aide un item « *Tests proposés par le système* »

✓ Aides, Commentaires et Solution
1 – Tests proposés par le système
Solution du système

Ces tests provoquent une exécution automatique de l'algorithme avec différents jeux de données critiques. Ainsi, il est facile de vérifier pour chaque cas, que le résultat obtenu est bien le résultat attendu.

6 Comment concevoir un algorithme ?

Ce qui précède s'est contenté de décrire les outils pour écrire un algorithme, mais ne dit rien sur la façon de concevoir un algorithme.

6.1 Il n'existe pas de procédé général

D'abord, il faut bien comprendre qu'il n'existe pas de méthode mécanique et infaillible pour concevoir un algorithme quelconque (cette affirmation est même un théorème de logique formelle). La conception d'un algorithme est une activité de construction progressive, souvent par essais et erreurs. Les étapes de l'algorithme ne sont pas forcément conçues dans l'ordre où elles sont finalement écrites.

Il ne faut pas rester paralysé par cette situation. Il faut essayer quelque chose, observer éventuellement la trace pour comprendre ce qui se passe, compléter, rectifier, effacer, recommencer.

Bien sûr, il ne faut pas faire tout à fait n'importe quoi, et l'expérience permet petit à petit d'acquérir certains réflexes et de reconnaître certains cas classiques, mais encore une fois ce serait une idée fausse de croire qu'on peut concevoir un algorithme du début à la fin en prévoyant toutes les étapes correctes dans le bon ordre.

D'autre part, il existe souvent plusieurs algorithmes possibles pour résoudre un exercice donné. Le logiciel permet d'en écrire certains (par le choix des variables et des expressions qu'il propose) mais pas forcément tous. Quand vous pensez ne pas pouvoir écrire l'algorithme que vous imaginez, essayez-en un autre. Il y a toujours au moins une solution.

6.2 Concevoir et analyser un programme avec des assertions

La méthode de base pour réfléchir à un algorithme est celle dite des « assertions ».

Elle consiste à se poser la question suivante pour chaque instruction écrite dans le texte : quand le programme s'exécutera et commencera à exécuter cette instruction, quelles propriétés posséderont les variables à ce moment-là (ce sont les *pré-assertions*) ? De même quand le programme terminera d'exécuter cette instruction (les propriétés sont les *post-assertions*).

On peut également considérer des *assertions d'évolution*, qui indiquent comment évoluent les variables entre le début d'une instruction et la fin d'une instruction (par exemple : telle variable a une valeur qui augmente).

Exemple : affectation

```
Poser K = K + 1
```

Si une pré-assertion est « K est un entier avec $K < N$ », alors une post-assertion est « K est un entier avec $K \leqslant N$ ».

Une assertion d'évolution est : « La valeur finale de K est strictement supérieure à sa valeur initiale »

Exemple : test

```
Si N > 0 faire :
   Poser V = N - 1
 sinon faire :
   Poser V = 0
```

Si une pré-assertion est : « N contient un nombre entier relatif », alors une post-assertion est : « V contient un nombre entier naturel », ce qu'on peut prouver avec un raisonnement par cas simple.

Si une pré-assertion est « N contient un entier naturel », alors une post-assertion est « V contient un entier naturel inférieur ou égal à N »

Exemple : assertion invariante

```
Poser K = K + 1
Poser S = S + K
```

Si une pré-assertion est « $S = 1 + \cdots + K$ », alors après les deux instructions une post-assertion est « $S = 1 + \cdots + K$ », c'est-à-dire la même assertion, qui est donc invariante.

Cela ne veut pas dire que l'état n'a pas évolué, cela veut dire que la relation entre S et K est la même. K et S ont augmenté (assertion d'évolution), mais on a toujours $S = 1 + \cdots + K$.

Si K vaut 2 au début et $S = 1 + 2 = 3$,

alors à la fin $K = 3$ et $S = 1 + 2 + 3 = 6$.

Cela est analogue à l'*hérédité* dans un raisonnement par récurrence :

Soit u une suite vérifiant $u_{n+1} = u_n + n + 1$, alors on peut démontrer que la propriété « $u_n = 1 + \cdots + n$ » est héréditaire.

Exemple : boucle

```
Tant que K < N faire :
    Poser K = K + 1
    Poser S = S + K
```

Si une pré-assertion est :

« K et N sont deux nombres entiers avec $K \leqslant N$ »,
alors une post-assertion est : « $K = N$ ». En effet, K augmente strictement à chaque tour de boucle à cause de l'instruction `Poser K = K + 1` (assertion d'évolution) et comme au départ on avait $K \leqslant N$, K se rapproche de N à chaque tour de boucle. On sait qu'avec des nombres entiers cela ne peut pas continuer indéfiniment, et il arrivera un moment où K deviendra égal à N. Alors la condition de continuation ($K < N$) sera fausse et la boucle s'arrêtera. En sortant de la boucle, on aura donc $K = N$.

6.3 Exemple : algorithme complet

```
Soit un entier N > 0
Poser K = 1
Poser S = 1
Tant que K < N faire :
    Poser K = K + 1
    Poser S = S + K
Afficher S
```

En combinant les différents raisonnements sur les assertions vus précédemment, on peut démontrer que la valeur finale de S qui sera affichée sera bien $1 + \cdots + N$.

Voici à nouveau l'algorithme complet, avec les assertions utilisées en bleu :

```
Soit un entier N > 0
N  est un entier > 0
Poser K = 1
Poser S = 1
S = 1 + ... + K  et  K  est inférieur ou égal à  N
Tant que K < N faire :
    S = 1 + ... + K  et  K < N
    Poser K = K + 1
    Poser S = S + K
    S = 1 + ... + K  et  K  est inférieur ou égal à  N
    de plus  K  a augmenté strictement
Après la boucle S = 1 + ... + K et K = N  donc S = 1 + ... + N
Afficher S
```

Le raisonnement global est un raisonnement par récurrence.
L'initialisation se fait avant la boucle, et la boucle assure l'hérédité.